Secure Packager and Encoder Key Exchange API Specification

Partner and customer guide

Secure Packager and Encoder Key Exchange API Specification: Partner and customer guide

Table of Contents

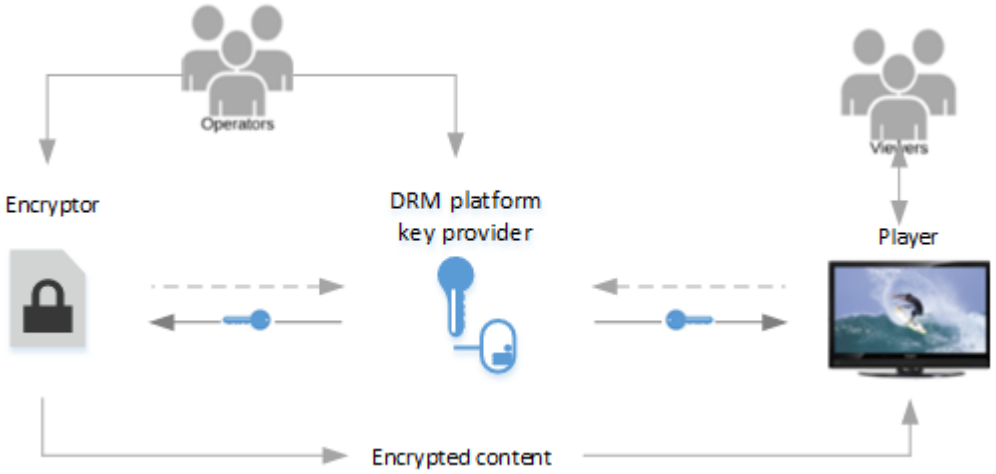
What is Secure Packager and Encoder Key Exchange?

Secure Packager and Encoder Key Exchange (SPEKE) defines the standard for communication between encryptors and packagers of media content and digital rights management (DRM) key providers. The specification accommodates encryptors running on-premises and in the AWS Cloud.

Topics

General architecture

The following illustration shows a high-level view of the SPEKE content encryption architecture for on-premises products.

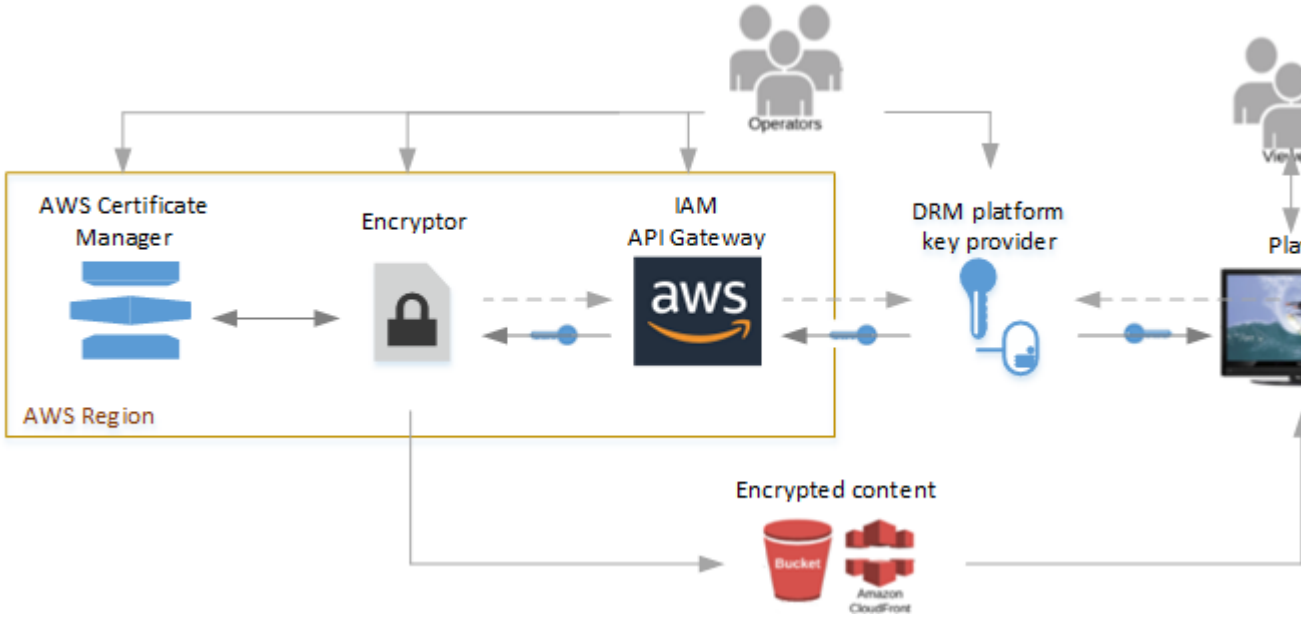


These are the main components of the preceding architecture:

- **Encryptor** – Provides the encryption technology. Receives encryption requests from its operator, and retrieves the required keys from the DRM key provider to secure the encrypted content.
- **DRM platform key provider** – Provides encryption keys to the encryptor through a SPEKE-compliant API. The provider also provides licenses to media players for decryption.
- **Player** – Requests keys from the same DRM platform key provider, which the player uses to unlock the content and serve it to its viewers.

AWS cloud-based architecture

The following illustration shows the high-level architecture when SPEKE is used with services and features running in the AWS Cloud.



These are the main services and components:

- **Encryptor** – Provides the encryption technology in the AWS Cloud. The encryptor receives requests from its operator and retrieves the required encryption keys from the DRM key provider, through Amazon API Gateway, to secure the encrypted content. It delivers the encrypted content to an Amazon S3 bucket or through an Amazon CloudFront distribution.
- **AWS IAM and Amazon API Gateway** – Manages customer-trusted roles and proxy communication between the encryptor and the key provider. API Gateway provides logging capabilities and lets customers control their relationships with the encryptor and with the DRM platform. Customers enable key provider access through IAM role configuration. The API Gateway must reside in the same AWS Region as the encryptor.
- **AWS Certificate Manager** – (Optional) Provides certificate management for content key encryption. Encrypting content keys is the recommended practice for secure communication. The certificate manager must reside in the same AWS Region as the encryptor.
- **DRM platform key provider** – Provides encryption keys to the encryptor through a SPEKE-compliant API. The provider also provides licenses to media players for decryption.
- **Player** – Requests keys from the same DRM platform key provider, which the player uses to unlock the content and serve it to its viewers.

How to get started

For additional introductory material about SPEKE, see Are you new to SPEKE? (p. 3).

Are you a customer?

Partner with an AWS Elemental DRM platform provider to get set up to use encryption. For details, see Customer onboarding (p. 5).

Are you a DRM platform provider or a customer with your own key provider?

Expose a REST API for your key provider in compliance with the SPEKE specification. For details, see SPEKE API specification (p. 7).

Are you new to SPEKE?

This section provides introductory material, for readers who are new to Secure Packager and Encoder Key Exchange (SPEKE).

For an introduction to SPEKE, watch the following webcast:

Related services and specifications

- API gateway permissions – How to control access to an API with AWS Identity and Access Management (AWS IAM) permissions.
- AWS AssumeRole – How to use AWS Security Token Service (AWS STS) to assume role functionality.
- AWS Sigv4 – How to sign an HTTP request using Signature Version 4.
- DASH-IF CPIX specification v2.0 – The DASH-IF Content Protection Information Exchange Format (CPIX) specification version, which this SPEKE v1.0 specification is based on.

- DASH-IF CPIX specification v2.3 – The DASH-IF Content Protection Information Exchange Format (CPIX) specification version, which this SPEKE v2.0 specification is based on.
- DASH-IF system IDs – The list of registered identifiers for DRM systems.
- https://github.com/awslabs/speke-reference-server – Example reference key provider to use with your AWS account, to help you get started with a SPEKE implementation in AWS.

Terminology

The following list defines the terminology used in this specification. Where possible, this specification follows the terminology used in the DASH-IF CPIX specification.

- **ARN** – Amazon Resource Name. Uniquely identifies an AWS resource.
- **Content key** – A cryptographic key used for encrypting part of the content.
- **Content provider** – A publisher who provides the rights and rules for delivering protected media. The content provider might also provide source media (mezzanine format, for transcoding), asset identifiers, key identifiers (KIDs), key values, encoding instructions, and content description metadata.
- **DRM** – Digital rights management. Used to protect copyrighted digital content from unapproved access.
- **DRM platform** – A system that provides DRM functionality and support to content encryptors and viewers, including providing DRM keys and licensing for content encryption and decryption.
- **DRM provider** – See DRM platform.
- **DRM system** – A standard for DRM implementations. Common DRM systems include Apple FairPlay, Google Widevine, and Microsoft PlayReady. DRM systems are used by content providers to secure digital content for delivery to viewers and for access by viewers. For a list of DRM systems that are registered with DASH-IF, see DASH-IF system IDs. The DASH-IF CPIX specification uses the term "DRM system" as defined here and, in some places, it uses "DRM system" to mean what this specification refers to as a DRM platform.
- **DRM solution** – See DRM platform.
- **DRM technology** – See DRM system.
- **Encryptor** – A media processing component that encrypts media content using keys obtained from the key provider. Encryptors typically also add DRM encryption signaling and metadata to the media.. Encryptors are usually encoders, packagers, and transcoders.
- **Key provider** – The component of a DRM platform that exposes a SPEKE REST API to handle key requests. The key provider might be the key server itself, or it might be another component of the platform.
- **Key server** – The component of a DRM platform that maintains keys for content encryption and decryption.
- **Operator** – A person in charge of operating the overall system, including the encryptor and the key provider.
- **Player** – A media player operating on behalf of a viewer. Gets its information from different sources, including the media manifest files, media files, and DRM licenses. Requests licenses from the DRM platform on behalf of the viewers.

Customer onboarding

Protect your content from unauthorized use by combining a Secure Packager and Encoder Key Exchange (SPEKE) digital rights management (DRM) key provider with your encryptor and with your media players. SPEKE defines the standard for communication between encryptors and packagers of media content and digital rights management (DRM) key providers. To onboard, you choose a DRM platform key provider and configure the communication between the key provider and your encryptors and players.

Topics

Get on board with a DRM platform provider

The following Amazon partners provide third-party DRM platform implementations for SPEKE. For details about their offerings and information about how to contact them, follow the links to their Amazon Partner Network pages. Partners that don't have a link don't currently have an Amazon Partner Network page, but you can contact them directly. The partners can help you get set up to use their platforms.

DRM Platform Provider	SPEKE v1 support	SPEKE v2 support (AWS Elemental MediaPackage)
Axinom	√	√
BuyDRM	√	√
castLabs	√	√
EZDRM	√	√
INKA Entworks	√	√
Insys Video Technologies	√	
Intertrust Technologies	√	√
Irdeto	√	√
JW Player	√	√
Kaltura	√	
NAGRA	√	√
NEXTSCAPE, Inc.	√	
SeaChange	√	
Verimatrix	√	
Viaccess-Orca	√	
WebStream	√	

SPEKE Support in AWS services and products

This section lists the SPEKE support that is provided by AWS Media Services that run in the AWS Cloud and by AWS on-premises media products. These services and products are the encryptors in the SPEKE content encryption architecture. Verify that your streaming protocol and the DRM system that you want are available for your service or product.

AWS Service or Product	SPEKE v1 support	SPEKE v2 support	Supported DRM Technologies
AWS Elemental MediaConvert - Service That Runs in the AWS Cloud	√		**Documentation**
AWS Elemental MediaPackage - Service That Runs in the AWS Cloud	√	√	**Documentation**
AWS Elemental Live - On-premises Product	√		**Documentation: MPEG-DASH / HLS**
AWS Elemental Server - On-premises Product	√		**Documentation**

SPEKE API specification

This is the REST API specification for Secure Packager and Encoder Key Exchange (SPEKE). Use this specification to provide DRM copyright protection for customers who use encryption.

In a video streaming workflow, the encryption engine communicates with the DRM platform key provider to request content keys. These keys are highly sensitive, so it is critical that the key provider and encryption engine establish a highly secure, trusted communication channel. You can also encrypt the content keys in the document for more secure, end-to-end encryption.

This specification addresses the following goals:

- Define a simple, trusted, highly secure interface that DRM vendors and customers can use to integrate with encryptors when content encryption is required.
- Cover VOD and live workflows, and include the error conditions and the authentication mechanisms that are required for robust, highly secure communication between encryptors and DRM key provider endpoints.
- Include support for HLS, MSS, and DASH packaging and their common DRM systems: FairPlay, PlayReady, and Widevine/CENC.
- Keep the specification simple and extensible, to support future DRM systems.
- Use a simple REST API.

Topics

Authentication

SPEKE requires authentication for on-premises products and for services and features that run in the AWS Cloud.

Topics

Authentication for AWS cloud implementations

SPEKE requires AWS authentication through IAM roles for use with an encryptor. IAM roles are created by the DRM provider or by the operator who owns the DRM endpoint in an AWS account. Each role is assigned an Amazon Resource Name (ARN), which the AWS Elemental service operator provides on the service console when requesting encryption. The role's policy permissions must be configured to give permission to access the key provider API and no other AWS resource access. When the encryptor contacts the DRM key provider, it uses the role ARN to assume the role of the key provider account holder, which returns temporary credentials for the encryptor to use to access the key provider.

One common implementation is for the operator or DRM platform vendor to use Amazon API Gateway in front of the key provider, and then enable AWS Identity and Access Management (AWS IAM) authorization on the API Gateway resource. You can use the following policy definition example and attach it to a new role to give permissions to the appropriate resource. In this case, the permissions are for all API Gateway resources:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-west-2:*:*/*/GET/*"
            ]
        }
    ]
}
```

Finally, the role requires the addition of a trust relationship, and the operator must be able to select the service.

The following example shows a role ARN that is created for accessing the DRM key provider:

```
arn:aws:iam::2949266363526:role/DRMKeyServer
```

For more information about the creation of a role, see AWS AssumeRole. For more information about signing a request, see AWS Sigv4.

Authentication for on-premises products

For on-premises products, we recommend that you use SSL/TLS and digest authentication for the best security, but at a minimum you should use basic authentication over HTTPS.

Both types of authentication use the `Authorization` header in the HTTP request:

- **Digest authentication** – The authorization header consists of the identifier `Digest` followed by a series of values that authenticate the request. Specifically, a response value is generated through a series of MD5 hash functions that include a unique, one-time-use nonce from the server that is used to ensure that the password travels securely.

- **Basic authentication** – The authorization header consists of the identifier `Basic` followed by a base-64 encoded string that represents the user name and password, separated by a colon.

For information about basic and digest authentication, including detailed information about the header, see the Internet Engineering Task Force (IETF) specification RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication.

SPEKE API v1

To be SPEKE-compliant, your DRM key provider must expose the REST API described in this specification. The encryptor makes API calls to your key provider.

> **Note**
> The code examples in this specification are for illustration purposes only. You can't run the examples because they aren't part of a complete SPEKE implementation.

Secure Packager and Encoder Key Exchange uses the DASH Industry Forum Content Protection Information Exchange Format (DASH-IF-CPIX) data structure definition for key exchange, with some restrictions. DASH-IF-CPIX defines a schema to provide an extensible, multi-DRM exchange from the DRM platform to the encryptor. This enables content encryption for all adaptive bitrate packaging formats at the time of content compression and packaging. Adaptive bitrate packaging formats include HLS, DASH, and MSS.

For detailed information about the exchange format, see the DASH Industry Forum CPIX specification at https://dashif.org/docs/DASH-IF-CPIX-v2-0.pdf.

Topics

SPEKE API v1 - Customizations and constraints to the DASH-IF specification

The DASH-IF CPIX specification, https://dashif.org/docs/DASH-IF-CPIX-v2-0.pdf, supports a number of use cases and topologies. The SPEKE API specification adheres to the CPIX specification with the following customizations and constraints:

- SPEKE follows the Encryptor Consumer workflow.
- For encrypted content keys, SPEKE applies the following restrictions:
  - SPEKE doesn't support digital signature verification (XMLDSIG) for request or response payloads.
  - SPEKE requires 2048 RSA-based certificates.
- For rotating key workflows, SPEKE requires the `ContentKeyUsageRule` filter, `KeyPeriodFilter`. SPEKE ignores all other `ContentKeyUsageRule` settings.

- SPEKE omits the `UpdateHistoryItemList` functionality. If the list is present in the response, SPEKE ignores it.
- SPEKE supports key rotation. SPEKE uses only the `ContentKeyPeriod@index to track the key period.
- To support MSS PlayReady, SPEKE uses a custom parameter under the `DRMSystem` tag, `SPEKE:ProtectionHeader`.
- For HLS packaging, if the `URIExtXKey` is present in the response, then it must contain the full data to add in the URI parameter of the `EXT-X-KEY` tag of an HLS playlist, with no further signaling requirement.
- For HLS playlist, under the `DRMSystem` tag, SPEKE provides the optional custom parameters `speke:KeyFormat` and `speke:KeyFormatVersions`, for the values of the `KEYFORMAT` and `KEYFORMATVERSIONS` parameters of the `EXT-X-KEY` tag.

  The HLS initialization vector (IV) always follows segment number unless explicitly specified by the operator.
- When requesting keys, the encryptor might use the optional `@explicitIV` attribute on the `ContentKey` element. The key provider can respond with an IV using `@explicitIV`, even if the attribute is not included in the request.
- The encryptor creates the key identifier (`KID`), which stays the same for any given content ID and key period. The key provider includes the `KID` in its response to the request document.
- The key provider might include a value for the `Speke-User-Agent` response header, to identify itself for debugging purposes.
- SPEKE does not currently support multiple tracks or keys per content.

  The SPEKE-compliant encryptor acts as a client and sends `POST` operations to the key provider endpoint. The encryptor might send a periodic `heartbeat` request to ensure that the connection between the encryptor and the key provider endpoint is healthy.

SPEKE API v1 - Standard payload components

In any SPEKE request, the encryptor can request responses for one or more DRM systems. The encryptor specifies the DRM systems in `<cpix:DRMSystemList>` of the request payload. Each system specification includes the key and indicates the type of response to return.

The following example shows a DRM system list with a single DRM system specification:

```
<cpix:DRMSystemList>
    <!--HLS AES-128 (systemId is implementation specific)-->
    <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
    systemId="81376844-f976-481e-a84e-cc25d39b0b33">
        <cpix:URIExtXKey></cpix:URIExtXKey>
        <speke:KeyFormat></speke:KeyFormat>
        <speke:KeyFormatVersions></speke:KeyFormatVersions>
    </cpix:DRMSystem>
</cpix:DRMSystemList>
```

The following table lists the main components of each `<cpix:DRMSystem>`.

Identifier	Description
`systemId` or `schemeId`	Unique identifier for the DRM system type, as registered with the DASH IF organization. For a list, see DASH-IF System IDs.

Identifier	Description
`kid`	The key ID. This is not the actual key, but an identifier that points to the key in a hash table.
`<cpix:UriExtXKey>`	Requests a standard unencrypted key. The key response type must be either this or the `PSSH` response.
`<cpix:PSSH>`	Requests a Protection System Specific Header (PSSH). This type of header contains a reference to the `kid`, the `systemID`, plus custom data for the DRM vendor, as part of Common Encryption (CENC). The key response type must be either this or the `UriExtXKey` response.

_Example Requests for Standard Key and for PSSH _

The following example shows part of a sample request from the encryptor to the DRM key provider, with the main components highlighted. The first request is for a standard key, while the second request is for a PSSH response:

```
<cpix:CPIX id="abc123" xmlns:cpix="urn:dashif:org:cpix"
xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:ama
    <cpix:ContentKeyList>
        <cpix:ContentKey kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
        explicitIV="OFj2IjCsPJFfMAxmQxLGPw==">></cpix:ContentKey>
    </cpix:ContentKeyList>
    <cpix:DRMSystemList>
        <!-- HLS AES-128 (systemId is implementation specific)-->
        <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
        systemId="81376844-f976-481e-a84e-cc25d39b0b33">
            <cpix:URIExtXKey></cpix:URIExtXKey>
            <speke:KeyFormat></speke:KeyFormat>
            <speke:KeyFormatVersions></speke:KeyFormatVersions>
        </cpix:DRMSystem>

        <!-- Common encryption (Widevine)-->
        <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
        systemId="edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
            <cpix:PSSH></cpix:PSSH>
        </cpix:DRMSystem>

    </cpix:DRMSystemList>
    ...
</cpix:CPIX>
```

System Id

request Key

System Id

request PSSH

_Example Responses for Standard Key and for PSSH _

The following example shows the corresponding response from the DRM key provider to the encryptor:

```
<cpix:CPIX xmlns:cpix="urn:dashif:org:cpix" xmlns:pskc="urn:ietf:params:x
xmlns:speke="urn:aws:amazon:com:speke" id="abc123">
    <cpix:ContentKeyList>
        <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw=="
        kid="98ee5596-cd3e-a20d-163a-e382420c6eff">
            <cpix:Data>
                <pskc:Secret>
                    <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:Plain
                </pskc:Secret>
            </cpix:Data>
        </cpix:ContentKey>
    </cpix:ContentKeyList>
    <cpix:DRMSystemList>
        <!-- HLS AES-128 (systemId is implementation specific) -->
        <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
        systemId="81376844-f976-481e-a84e-cc25d39b0b33">  <-- System Id

            <cpix:URIExtXKey>aHR0cHM6Ly83azR5dHV4cTVkLmV4ZWN1dGUtYXBpLnVz
            uY29tL0VrZVN0YWdlL2NsaWVudC9hYmMxMjMvOThlZTU1OTYtY2QzZS1hMjBk
            m</cpix:URIExtXKey>
            <speke:KeyFormat>aWRlbnRpdHk=</speke:KeyFormat>
            <speke:KeyFormatVersions>MQ==</speke:KeyFormatVersions>
        </cpix:DRMSystem>

        <!-- Common encryption (Widevine) -->
        <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
        systemId="edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">  <-- System Id

            <cpix:PSSH>AAAAanBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAAEoIARIQe
            2lkZXZpbmVfdGVzdCIfa2V5LWlkOmVTSWNibGFOYmI3RGppNnNBdEtaelE9PS
        </cpix:DRMSystem>

    </cpix:DRMSystemList>
    ...
</cpix:CPIX>
```

SPEKE API v1 - Live workflow method call examples

Request Syntax Example

The following URL is an example and does not indicate a fixed format:

```
POST https://speke-compatible-server/speke/v1.0/copyProtection
```

Request Body

A CPIX element.

Request Headers

Name	Type	Occurs	Description
`AWS Authorization`	String	1..1	See AWS Sigv4
`X-Amz-Security-Token`	String	1..1	See AWS Sigv4
`X-Amz-Date`	String	1..1	See AWS Sigv4
`Content-Type`	String	1..1	application/xml

Response Headers

Name	Type	Occurs	Description
`Speke-User-Agent`	String	1..1	String that identifies the key provider
`Content-Type`	String	1..1	application/xml

Request Response

HTTP CODE	Payload Name	Occurs	Description
`200 (Success)`	CPIX	1..1	DASH-CPIX payload response
`4XX (Client error)`	Client error message	1..1	Description of the client error
`5XX (Server error)`	Server error message	1..1	Description of the server error

Note
The examples in this section do not include content key encryption. For information about how to add content key encryption, see Content key encryption (p. 18).

Live Example Request Payload with Keys in the Clear

The following example shows a typical live request payload from the encryptor to the DRM key provider:

```
<cpix:CPIX id="abc123" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke">
 <cpix:ContentKeyList>
  <cpix:ContentKey kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 explicitIV="OFj2IjCsPJFfMAxmQxLGPw==">></cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- HLS AES-128 (systemId is implementation specific)-->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="81376844-f976-481e-
a84e-cc25d39b0b33">
   <cpix:URIExtXKey></cpix:URIExtXKey>
   <speke:KeyFormat></speke:KeyFormat>
   <speke:KeyFormatVersions></speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- HLS SAMPLE-AES -->
```

```
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:URIExtXKey></cpix:URIExtXKey>
   <speke:KeyFormat></speke:KeyFormat>
   <speke:KeyFormatVersions></speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- Common encryption (Widevine)-->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>

  <!-- Common encryption / MSS (Playready) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <speke:ProtectionHeader></speke:ProtectionHeader>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyPeriodList>
  <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
 </cpix:ContentKeyPeriodList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

Live Example Response Payload with Keys in the Clear

The following example shows a typical response payload from the DRM key provider:

```
<cpix:CPIX xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke"
 id="abc123">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- HLS AES-128 (systemId is implementation specific) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="81376844-f976-481e-
a84e-cc25d39b0b33">

 <cpix:URIExtXKey>aHR0cHM6Ly83azR5dHV4cTVkLmV4ZWN1dGUtYXBpLnVzLXdlc3QtMi5hbWF6b25hd3MuY29tL0VrZVN0YWdlI
cpix:URIExtXKey>
   <speke:KeyFormat>aWRlbnRpdHk=</speke:KeyFormat>
   <speke:KeyFormatVersions>MQ==</speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- HLS SAMPLE-AES -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">

 <cpix:URIExtXKey>aHR0cHM6Ly83azR5dHV4cTVkLmV4ZWN1dGUtYXBpLnVzLXdlc3QtMi5hbWF6b25hd3MuY29tL0VrZVN0YWdlI
cpix:URIExtXKey>
   <speke:KeyFormat>Y29tLmFwcGxlLnN0cmVhbWluZ2tleWRlbGl2ZXJ5</speke:KeyFormat>
```

```
   <speke:KeyFormatVersions>MQ==</speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- Common encryption (Widevine) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:PSSH>AAAAanBzc2gAAAAA7e
+LqXnWSs6jyCfc1R0h7QAAAEoIARIQeSIcblaNbb7Dji6sAtKZzRoNd2lkZXZpbmVfdGVzdCIfa2V5LWlkOmVTSWNibGFOYmI3RGppN
cpix:PSSH>
  </cpix:DRMSystem>

  <!-- Common encryption / MSS (Playready) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">

 <speke:ProtectionHeader>CgMAAAEAAQAAAzwAVwBSAE0ASABFAEEARABFAFIAIAB4AG0AbABuAHMAPQAiAGgAdAB0AHAAOgAvAC
+ADwAQQBMAEcASQBEAD4AQQBFAFMAQwBUAFIAPAAvAEEATABHAEkARAA
+ADwALwBQAFIATwBUAEUAQwBUAEkATgBGAE8APgA8AEsASQBEAD4ATwBXAGoAaAB0AHIAMwB1ADkAawArAHIAZABvADEASQBMAFkAMA
+AGgAdAB0AHAAOgAvAC8AcABsAGEAeQByAGUAYQBkAHkALgBkAGkAcgBlAGMAdAB0AGEAcABzAC4AbgBlAHQALwBwAHIALwBzAHYAYw
+ADwALwBXAFIATQBIAEUAQQBEAEUAUgA+AA==</speke:ProtectionHeader>

 <cpix:PSSH>AAADMHBzc2gAAAAAmgTweZhAQoarkuZb4IhflQAAAxAQAwAAAQABAAYDPABXAFIATQBIAEUAQQBEAEUAUgAgAHgAbQB
+ADwASwBFAFkATABFAE4APgAxADYAPAAvAEsARQBZAEwARQBOAD4APABBAEwARwBJAEQAPgBBAEUAUwBDAFQAUgA8AC8AQQBMAEcASQ
+ADwASwBJAEQAPgBiAGgAdwBpAGUAWQAxAFcAdgBtADMARABqAGkANgBzAEEAdABLAFoAegBRAD0APQA8AC8ASwBJAEQAPgA8AEMASA
+AGEAVABtAFAASgBWAEMAVgBaADYAcwA9ADwALwBDAEgARQBDAEsAUwBVAE0APgA8AEwAQQBfAFUAUgBMAD4AaAB0AHQAcABzADoALw
+ADwALwBEAEEAVABBAD4APAAvAFcAUgBNAEgARQBBAEQARQBSAD4A</cpix:PSSH>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyPeriodList>
  <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
 </cpix:ContentKeyPeriodList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

SPEKE API v1 - VOD workflow method call examples

Request Syntax Example

The following URL is an example and does not indicate a fixed format.

```
POST https://speke-compatible-server/speke/v1.0/copyProtection
```

Request Body

A CPIX element.

Response Headers

Name	Type	Occurs	Description
`Speke-User-Agent`	String	1..1	String that identifies the key provider
`Content-Type`	String	1..1	application/xml

Request Response

HTTP CODE	Payload Name	Occurs	Description
`200 (Success)`	CPIX	1..1	DASH-CPIX payload response
`4XX (Client error)`	Client error message	1..1	Description of the client error
`5XX (Server error)`	Server error message	1..1	Description of the server error

Note
The examples in this section do not include content key encryption. For information on how to add content key encryption, see Content key encryption (p. 18).

VOD Example Request Payload with Keys in the Clear

The following example shows a basic VOD request payload from the encryptor to the DRM key provider:

```
<cpix:CPIX id="abc123" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke">
 <cpix:ContentKeyList>
  <cpix:ContentKey kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 explicitIV="OFj2IjCsPJFfMAxmQxLGPw==">></cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- HLS AES-128 (systemId is implementation specific)-->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="81376844-f976-481e-
a84e-cc25d39b0b33">
   <cpix:URIExtXKey></cpix:URIExtXKey>
   <speke:KeyFormat></speke:KeyFormat>
   <speke:KeyFormatVersions></speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- HLS SAMPLE-AES -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:URIExtXKey></cpix:URIExtXKey>
   <speke:KeyFormat></speke:KeyFormat>
   <speke:KeyFormatVersions></speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- Common encryption (Widevine)-->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>

  <!-- Common encryption / MSS (Playready) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <speke:ProtectionHeader></speke:ProtectionHeader>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
    </cpix:CPIX>
```

VOD Example Response Payload with Keys in the Clear

The following example shows a basic VOD response payload from the DRM key provider:

```
<cpix:CPIX xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke"
 id="abc123">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- HLS AES-128 (systemId is implementation specific) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="81376844-f976-481e-
a84e-cc25d39b0b33">

 <cpix:URIExtXKey>aHR0cHM6Ly83azR5dHV4cTVkLmV4ZWN1dGUtYXBpLnVzLXdlc3QtMi5hbWF6b25hd3MuY29tL0VrZVN0YWdlI
cpix:URIExtXKey>
   <speke:KeyFormat>aWRlbnRpdHk=</speke:KeyFormat>
   <speke:KeyFormatVersions>MQ==</speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- HLS SAMPLE-AES -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">

 <cpix:URIExtXKey>aHR0cHM6Ly83azR5dHV4cTVkLmV4ZWN1dGUtYXBpLnVzLXdlc3QtMi5hbWF6b25hd3MuY29tL0VrZVN0YWdlI
cpix:URIExtXKey>
   <speke:KeyFormat>Y29tLmFwcGxlLnN0cmVhbWluZ2tleWRlbGl2ZXJ5</speke:KeyFormat>
   <speke:KeyFormatVersions>MQ==</speke:KeyFormatVersions>
  </cpix:DRMSystem>

  <!-- Common encryption (Widevine) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:PSSH>AAAAanBzc2gAAAAA7e
+LqXnWSs6jyCfc1R0h7QAAAEoIARIQeSIcblaNbb7Dji6sAtKZzRoNd2lkZXZpbmVfdGVzdCIfa2V5LWlkOmVTSWNibGFOYmI3RGppN
cpix:PSSH>
  </cpix:DRMSystem>

  <!-- Common encryption / MSS (Playready) -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">

 <speke:ProtectionHeader>CgMAAAEAAQAAAzwAVwBSAE0ASABFAEEARABFAFIAIAB4AG0AbABuAHMAPQAiAGgAdAB0AHAAOgAvAC
+ADwAQQBMAEcASQBEAD4AQQBFAFMAQwBUAFIAPAAvAEEATABHAEkARAA
+ADwALwBQAFIATwBUAEUAQwBUAEkATgBGAE8APgA8AEsASQBEAD4ATwBXAGoAaAB0AHIAMwB1ADkAawArAHIAZABvADEASQBMAFkAMA
+AGgAdAB0AHAAOgAvAC8AcABsAGEAeQByAGUAYQBkAHkALgBkAGkAcgBlAGMAdAB0AGEAcABzAC4AbgBlAHQALwBwAHIALwBzAHYAYw
+ADwALwBXAFIATQBIAEUAQQBEAEUAUgA+AA==</speke:ProtectionHeader>

 <cpix:PSSH>AAADMHBzc2gAAAAAmgTweZhAQoarkuZb4IhflQAAAxAQAwAAAQABAAYDPABXAFIATQBIAEUAQQBEAEUAUgAgAHgAbQB
+ADwASwBFAFkATABFAE4APgAxADYAPAAvAEsARQBZAEwARQBOAD4APABBAEwARwBJAEQAPgBBAEUAUwBDAFQAUgA8AC8AQQBMAEcASQ
+ADwASwBJAEQAPgBiAGgAdwBpAGUAWQAxAFcAdgBtADMARABqAGkANgBzAEEAdABLAFoAegBRAD0APQA8AC8ASwBJAEQAPgA8AEMASA
+AGEAVABtAFAASgBWAEMAVgBaADYAcwA9ADwALwBDAEgARQBDAEsAUwBVAE0APgA8AEwAQQBfAFUAUgBMAD4AaAB0AHQAcABzADoALw
+ADwALwBEAEEAVABBAD4APAAvAFcAUgBNAEgARQBBAEQARQBSAD4A</cpix:PSSH>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
</cpix:CPIX>
```

SPEKE API v1 - Content key encryption

You can optionally add content key encryption to your SPEKE implementation. Content key encryption guarantees full end-to-end protection by encrypting the content keys for transit, in addition to encrypting the content itself. If you don't implement this for your key provider, you rely on the transport layer encryption plus strong authentication for security.

To use content key encryption for encryptors running in AWS Cloud, customers import certificates into the AWS Certificate Manager and then use the resulting certificate ARNs for their encryption activities. The encryptor uses the certificate ARNs and the ACM service to provide encrypted content keys to the DRM key provider.

Restrictions

SPEKE supports content key encryption as specified in the DASH-IF CPIX specification with the following restrictions:

- SPEKE doesn't support digital signature verification (XMLDSIG) for request or response payloads.
- SPEKE requires 2048 RSA-based certificates.

These restrictions are also listed in Customizations and constraints to the DASH-IF specification (p. 9).

Implement content key encryption

To provide content key encryption, include the following in your DRM key provider implementations:

- Handle the element `<cpix:DeliveryDataList>` in the request and response payloads.
- Provide encrypted values in the `<cpix:ContentKeyList>` of the response payloads.

For more information about these elements, see the DASH-IF CPIX 2.0 specification.

Example Content Key Encryption Element `<cpix:DeliveryDataList>` *in the Request Payload*

The following example highlights the added `<cpix:DeliveryDataList>` element in bold:

```
<?xml version="1.0" encoding="UTF-8"?>
<cpix:CPIX id="example-test-doc-encryption"
    xmlns:cpix="urn:dashif:org:cpix"
    xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc"
    xmlns:speke="urn:aws:amazon:com:speke">
    <cpix:DeliveryDataList>
        <cpix:DeliveryData id="<ORIGIN SERVER ID>">
            <cpix:DeliveryKey>
                <ds:X509Data>
                    <ds:X509Certificate><X.509 CERTIFICATE, BASE-64 ENCODED></
ds:X509Certificate>
                </ds:X509Data>
            </cpix:DeliveryKey>
        </cpix:DeliveryData>
    </cpix:DeliveryDataList>
    <cpix:ContentKeyList>
     ...
    </cpix:ContentKeyList>
</cpix:CPIX>
```

Example Content Key Encryption Element `<cpix:DeliveryDataList>` *in the Response Payload*

The following example highlights the added `<cpix:DeliveryDataList>` element in bold:

```
<cpix:CPIX xmlns:cpix="urn:dashif:org:cpix"
    xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc"
    xmlns:speke="urn:aws:amazon:com:speke" id="hls_test_001">
    <cpix:DeliveryDataList>
        <cpix:DeliveryData id="<ORIGIN SERVER ID>">
            <cpix:DeliveryKey>
                <ds:X509Data>
                    <ds:X509Certificate><X.509 CERTIFICATE, BASE-64 ENCODED></
ds:X509Certificate>
                </ds:X509Data>
            </cpix:DeliveryKey>
            <cpix:DocumentKey Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc">
                <cpix:Data>
                    <pskc:Secret>
                        <pskc:EncryptedValue>
                            <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#rsa-oaep-mgf1p" />
                            <enc:CipherData>
                                <enc:CipherValue><RSA CIPHER VALUE></enc:CipherValue>
                            </enc:CipherData>
                        </pskc:EncryptedValue>
                        <pskc:ValueMAC>qnei/5TsfUwDu+8bhsZrLjDRDngvmnUZD2eva7SfXWw=</
pskc:ValueMAC>
                    </pskc:Secret>
                </cpix:Data>
            </cpix:DocumentKey>
            <cpix:MACMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha512">
                <cpix:Key>
                    <pskc:EncryptedValue>
                        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#rsa-oaep-mgf1p" />
                        <enc:CipherData>
                            <enc:CipherValue><RSA CIPHER VALUE></enc:CipherValue>
                        </enc:CipherData>
                    </pskc:EncryptedValue>
                    <pskc:ValueMAC>DGqdpHUfFKxdsO9+EWrPjtdTCVfjPLwwtzEcFC/j0xY=</
pskc:ValueMAC>
                </cpix:Key>
            </cpix:MACMethod>
        </cpix:DeliveryData>
    </cpix:DeliveryDataList>
    <cpix:ContentKeyList>
     ...
    </cpix:ContentKeyList>
</cpix:CPIX>
```

Example Content Key Encryption Element `<cpix:ContentKeyList>` *in the Response Payload*

The following example shows encrypted content key handling in the `<cpix:ContentKeyList>` element of the response payload. This uses the `<pskc:EncryptedValue>` element:

```
   <cpix:ContentKeyList>
        <cpix:ContentKey kid="682681c8-69fa-4434-9f9f-1a7f5389ec02">
            <cpix:Data>
                <pskc:Secret>
                    <pskc:EncryptedValue>
                        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#aes256-cbc" />
                        <enc:CipherData>
                            <enc:CipherValue>NJYebfvJ2TdMm3k6v
+rLNVYb0NoTJoTLBBdbpe8nmilEfp82SKa7MkqTn2lmQBPB</enc:CipherValue>
                        </enc:CipherData>
                    </pskc:EncryptedValue>
```

```
                    <pskc:ValueMAC>t9lW4WCebfS1GP+dh0IicMs+2+jnrAmfDa4WU6VGHc4=</
pskc:ValueMAC>
                </pskc:Secret>
            </cpix:Data>
        </cpix:ContentKey>
    </cpix:ContentKeyList>
```

By comparison, the following example shows a similar response payload with the content key delivered unencrypted, as a clear key. This uses the `<pskc:PlainValue>` element:

```
    <cpix:ContentKeyList>
        <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw=="
 kid="682681c8-69fa-4434-9f9f-1a7f5389ec02">
            <cpix:Data>
                <pskc:Secret>
                    <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
                </pskc:Secret>
            </cpix:Data>
        </cpix:ContentKey>
    </cpix:ContentKeyList>
```

SPEKE API v1 - Heartbeat

Request Syntax Example

The following URL is an example and does not indicate a fixed format:

```
GET https://speke-compatible-server/speke/v1.0/heartbeat
```

Request Response

HTTP CODE	Payload Name	Occurs	Description
`200 (Success)`	statusMessage	1..1	Message that describes the status

SPEKE API v1 - Overriding the key identifier

The encryptor creates a new key identifier (KID) each time that it rotates keys. It passes the KID to the DRM key provider in its requests. Almost always, the key provider responds using the same KID, but it can provide a different value for the KID in the response.

The following is an example request with the KID `11111111-1111-1111-1111-111111111111`:

```
    <cpix:CPIX id="abc123" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke">
      <cpix:ContentKeyList>
       <cpix:ContentKey kid="11111111-1111-1111-1111-111111111111"></cpix:ContentKey>
      </cpix:ContentKeyList>
      <cpix:DRMSystemList>
       <!-- Common encryption (Widevine)-->
       <cpix:DRMSystem kid="11111111-1111-1111-1111-111111111111"
 systemId="edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
        <cpix:PSSH />
       </cpix:DRMSystem>
      </cpix:DRMSystemList>
      <cpix:ContentKeyPeriodList>
```

```
      <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"
 index="1" />
     </cpix:ContentKeyPeriodList>
     <cpix:ContentKeyUsageRuleList>
      <cpix:ContentKeyUsageRule kid="11111111-1111-1111-1111-111111111111">
       <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" />
      </cpix:ContentKeyUsageRule>
     </cpix:ContentKeyUsageRuleList>
    </cpix:CPIX>
```

The following response overrides the KID to `22222222-2222-2222-2222-222222222222`:

```
    <cpix:CPIX xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke"
 id="abc123">
     <cpix:ContentKeyList>
      <cpix:ContentKey explicitIV="ASgwx9pQ2/2lnDzJsUxWcQ=="
 kid="22222222-2222-2222-2222-222222222222">
       <cpix:Data>
        <pskc:Secret>
         <pskc:PlainValue>p3dWaHARtL97MpT7TE916w==</pskc:PlainValue>
        </pskc:Secret>
       </cpix:Data>
      </cpix:ContentKey>
     </cpix:ContentKeyList>
     <cpix:DRMSystemList>
      <cpix:DRMSystem kid="22222222-2222-2222-2222-222222222222"
 systemId="edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
       <cpix:PSSH>AAAAanBzc2gAAAAA7e
+LqXnWSs6jyCfc1R0h7QAAAEoIARIQeSIcblaNbb7Dji6sAtKZzRoNd2lkZXZpbmVfdGVzdCIfa2V5LWlkOmVTSWNibGFOYmI3RGppN
cpix:PSSH>
      </cpix:DRMSystem>
     </cpix:DRMSystemList>
     <cpix:ContentKeyPeriodList>
      <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"
 index="1" />
     </cpix:ContentKeyPeriodList>
     <cpix:ContentKeyUsageRuleList>
      <cpix:ContentKeyUsageRule kid="22222222-2222-2222-2222-222222222222">
       <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" />
      </cpix:ContentKeyUsageRule>
     </cpix:ContentKeyUsageRuleList>
    </cpix:CPIX>
```

SPEKE API v2

To be SPEKE-compliant, your DRM key provider must expose the REST API described in this specification. The encryptor makes API calls to your key provider.

> **Note**
> The code examples in this specification are for illustration purposes only. You can't run the examples because they aren't part of a complete SPEKE implementation.

Secure Packager and Encoder Key Exchange uses the DASH Industry Forum Content Protection Information Exchange Format (DASH-IF-CPIX) data structure definition for key exchange, with some restrictions. DASH-IF-CPIX defines a schema to provide an extensible, multi-DRM exchange from the DRM platform to the encryptor. This enables content encryption for all adaptive bitrate packaging formats at the time of content compression and packaging. Adaptive bitrate packaging formats include HLS, DASH, and MSS.

Starting with its version 2.0, SPEKE is aligned on a specific CPIX version:

On the SPEKE side, this is enforced through the use of the `X-Speke-Version` HTTP header, and on the CPIX side through the use of the `CPIX@version` attribute. A lack of these elements in the requests is typical of SPEKE v1 legacy workflows. In SPEKE v2 workflows, the key provider is expected to process CPIX documents only if it supports both version parameters.

For detailed information about the exchange format, see the DASH Industry Forum CPIX 2.3 specification.

Overall, SPEKE v2.0 brings the following evolutions compared to SPEKE v1.0:

- All tags from the SPEKE XML namespace are deprecated in favor of equivalent tags in the CPIX XML namespace
- `SPEKE:ProtectionHeader` is deprecated and replaced by `CPIX:DRMSystem.SmoothStreamingProtectionHeaderData`
- `CPIX:URIExtXKey`, `SPEKE:KeyFormat` and `SPEKE:KeyFormatVersions` are deprecated and replaced by `CPIX:DRMSystem.HLSSignalingData`
- `CPIX@id` is replaced by `CPIX@contentId`
- New mandatory CPIX attributes: `CPIX@version`, `ContentKey@commonEncryptionScheme`
- New optional CPIX element: `DRMSystem.ContentProtectionData`
- Support for multiple content keys
- Cross-versioning mechanism between SPEKE and CPIX
- HTTP headers evolution: new `X-Speke-Version` header, `Speke-User-Agent` header renamed into `X-Speke-User-Agent`
- Heartbeat API deprecation

As the SPEKE v1.0 specification stays unchanged, existing implementations don't need to change to continue supporting SPEKE v1.0 workflows.

Topics

SPEKE API v2 - Customizations and constraints to the DASH-IF specification

The DASH Industry Forum CPIX 2.3 specification supports a number of use cases and topologies. The SPEKE API v2.0 specification defines both a CPIX Profile and an API for CPIX. In order to achieve these two goals, it adheres to the CPIX specification with the following customizations and constraints:

CPIX Profile

- SPEKE follows the Encryptor Consumer workflow.
- For encrypted content keys, SPEKE applies the following restrictions:
  - SPEKE doesn't support digital signature verification (XMLDSIG) for request or response payloads.
  - SPEKE requires 2048 RSA-based certificates.

- SPEKE leverages only a subset of CPIX functionalities:
  - SPEKE omits the `UpdateHistoryItemList` functionality. If the list is present in the response, SPEKE ignores it.
  - SPEKE omits the root/leaf key functionality. If the `ContentKey@dependsOnKey` attribute is present in the response, SPEKE ignores it.
  - SPEKE omits the `BitrateFilter` element and the `VideoFilter@wcg` attribute. If these elements or attributes are present in the CPIX payload, SPEKE ignores it.
- Only the elements or attributes referenced as 'Supported' on the Standard Payload Components page (p. 24) or the Encryption contract page (p. 28) can be used in CPIX documents exchanged with SPEKE v2.
- When included in a CPIX request by the encryptor, all elements and attributes shall carry a valid value in the key provider CPIX response. If not, the encryptor shall stop and throw an error.
- SPEKE supports key rotation with `KeyPeriodFilter` elements. SPEKE uses only the `ContentKeyPeriod@index` to track the key period.
- For HLS signaling, multiple `DRMSystem.HLSSignalingData` elements must be used: one with a `DRMSystem.HLSSignalingData@playlist` attribute value of 'media', and another one with a `DRMSystem.HLSSignalingData@playlist` attribute value of 'master'.
- When requesting keys, the encryptor might use the optional `@explicitIV` attribute on the `ContentKey` element. The key provider can respond with an IV using `@explicitIV`, even if the attribute is not included in the request.
- The encryptor creates the key identifier (`KID`), which stays the same for any given content ID and key period. The key provider includes the `KID` in its response to the request document.
- The encryptor shall include a value for the `CPIX@contentId` attribute. When receiving an empty value for this attribute, the key provider shall return an error with description 'Missing CPIX@contentId'. `CPIX@contentId` value cannot be overriden by the key provider.

  `CPIX@id` value, if not null, shall be ignored by the key provider.
- The encryptor shall include a value for the `CPIX@version` attribute. When receiving an empty value for this attribute, the key provider shall return an error with description 'Missing CPIX@version'. When receiving a request with an unsupported version, the error description returned by the key provider shall be 'Unsupported CPIX@version'.

  `CPIX@version` value cannot be overriden by the key provider.
- The encryptor shall include a value for the `ContentKey@commonEncryptionScheme` attribute for each requested key. When receiving an empty value for this attribute, the key provider shall return an error with description 'Missing ContentKey@commonEncryptionScheme for KID `id` '.

  A unique CPIX document cannot mix multiple values for different `ContentKey@commonEncryptionScheme` attributes. When receiving such a combination, the key provider shall return an error with description 'Non compliant ContentKey@commonEncryptionScheme combination'.

  Not all `ContentKey@commonEncryptionScheme` values are compatible with all DRM technologies. When receiving such a combination, the key provider shall return an error with description 'ContentKey@commonEncryptionScheme non compatible with DRMSystem `id` '.

  `ContentKey@commonEncryptionScheme` value cannot be overriden by the key provider.
- When receiving different values for `DRMSystem@PSSH` and `DRMSystem.ContentProtectionData` innerXML `<pssh>` element in the CPIX response body, the encryptor shall stop and throw an error.

API for CPIX

- The key provider shall include a value for the `X-Speke-User-Agent` HTTP response header.
- A SPEKE-compliant encryptor acts as a client and sends POST operations to the key provider endpoint.

- The encryptor shall include a value for the `X-Speke-Version` HTTP request header, with the SPEKE version used with the request, formulated as MajorVersion.MinorVersion, like '2.0' for SPEKE v2.0. If the key provider doesn't support the SPEKE version used by the encryptor for the current request, the key provider shall return an error with description 'Unsupported SPEKE version' and not try to process the CPIX document on a best effort basis.

  The `X-Speke-Version` header value defined by the encryptor cannot be modified by the key provider in the response to the request.
- When receiving errors in the response body, the encryptor shall throw an error and not retry the request with a SPEKE v1.0 versioning.

  If the key provider doesn't return an error but fails to return a CPIX document that includes the mandatory information, the encryptor should stop and throw an error.

The following table summarizes the standard messages that must be returned by the key provider in the body of the message. The HTTP response code in error cases shall be a 4XX or a 5XX, never a 200. The 422 error code can be used for all errors related to SPEKE/CPIX.

Error case	Error message
CPIX@contentId is not defined	Missing CPIX@contentId
CPIX@version is not defined	Missing CPIX@version
CPIX@version is not supported	Unsupported CPIX@version
ContentKey@commonEncryptionScheme is not defined	Missing ContentKey@commonEncryptionScheme for KID `id` (where `id` equals ContentKey@kid value)
Multiple ContentKey@commonEncryptionScheme values used in a single CPIX document	Non compliant ContentKey@commonEncryptionScheme combination
ContentKey@commonEncryptionScheme is not compatible with DRM technology	ContentKey@commonEncryptionScheme non compatible with DRMSystem `id` (where `id` equals DRMSystem@systemId value)
X-Speke-Version header value is not a supported SPEKE version	Unsupported SPEKE version
The encryption contract is malformed	Malformed encryption contract
The encryption contract contradicts DRM security levels constraints	Requested CPIX encryption contract not supported
The encryption contract doesn't include any VideoFilter or AudioFilter element	Missing CPIX encryption contract

SPEKE API v2 - Standard payload components

Through a single SPEKE request, the encryptor can request multiple content keys, together with the necessary manfest signaling for multiple packaging formats, according to the encryption contract that is defined for a given content.

In order to cover all these aspects, a standard CPIX document is composed of three mandatory list sections, plus an optional list section for live content key rotation.

<cpix:ContentKeyList> section and top level <cpix:CPIX> element

This is a mandatory section, relevant for both Live and VOD streaming, defining the different content keys that need to be used by the encryptor. The `<cpix:ContentKeyList>` element can contain one or several `<cpix:ContentKey>` child elements, each of them describing a distinct content key.

As per the CPIX specification, the possible values of the `ContentKey@commonEncryptionScheme` attribute are defined in the Common encryption in ISO base media file format files specification (ISO/IEC 23001-7:2016):

- 'cenc': AES-CTR mode full sample and video NAL Subsample encryption
- 'cbc1': AES-CBC mode full sample and video NAL Subsample encryption
- 'cens': AES-CTR mode partial video NAL pattern encryption
- 'cbcs': AES-CBC mode partial video NAL pattern encryption

The following example shows a CPIX document with a single, non encrypted, content key:

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
 </cpix:ContentKeyList>
 ...
</cpix:CPIX>
```

By default, content keys are not encrypted, as in the example below. But the encryption of content keys can be requested by the encryptor through the inclusion of the <cpix:DeliveryDataList> element. Please refer to the Content Key Encryption section for more details.

Element supported by SPEKE	Mandatory attributes	Optional attributes	Mandatory child elements	Optional child elements
<cpix:CPIX>	contentId, version, xmlns:cpix, xmlns:pskc	name, xmlns:enc	one <cpix:ContentKeyList>, one <cpix:DRMSystemList>, one <cpix:ContentKeyUsageRuleList>	one <cpix:DeliveryDataList>, one <cpix:ContentKeyPeriodList>
<cpix:ContentKeyList>		id	at least one <cpix:ContentKey>	-
<cpix:ContentKey>	kid, commonEncryptionScheme, Data	id, Algorithm, explicitIV	one <pskc:Secret>	-
<pskc:Secret>	PlainValue or EncryptedValue	ValueMAC	-	<enc:EncryptionMethod>, <enc:CipherData>

<cpix:DRMSystemList> section

This is a mandatory section, relevant for both Live and VOD streaming, defining the different DRM systems that need to be leveraged together with the content keys.

The following example shows a DRM system list with a single PlayReady DRM system specification:

```
<cpix:DRMSystemList>
 <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
  <cpix:HLSSignalingData playlist="media">HicXmbZ2m[...]4==</cpix:HLSSignalingData>
  <cpix:HLSSignalingData playlist="master">HicXmbZ2m[...]jEi</cpix:HLSSignalingData>
  <cpix:ContentProtectionData>t7WwH24FI[...]YCC</cpix:ContentProtectionData>
  <cpix:PSSH>FFFFanBzc[...]A==</cpix:PSSH>
  <cpix:SmoothStreamingProtectionHeaderData>s5RrJ12HL[...]UBB</
cpix:SmoothStreamingProtectionHeaderData>
 </cpix:DRMSystem>
</cpix:DRMSystemList>
```

For a complete list of DRM systemIDs, please refer to the Content Protection section of the DASH-IF Identifiers repository.

Element supported by SPEKE	Mandatory attributes	Optional attributes	Mandatory child elements	Optional child elements
<cpix:DRMSystemList>		id	at least one <cpix:DRMSystem>	-
<cpix:DRMSystem>	kid, systemId	id, name, PSSH	-	ContentProtectionData, SmoothStreamingProtectionHea two <cpix:HLSSignalingData> elements with different playlist attribute value

`DRMSystem@PSSH` is mandatory if ISO-BMFF encapsulation is applied to media segments. `DRMSystem.ContentProtectionData` innerXML `<pssh>` element is leveraged by the encryptor only for manifest signaling purposes.

If `DRMSystem@PSSH` is present and `DRMSystem.ContentProtectionData` contains a innerXML `<pssh>` element, both values shall be identical.

If `DRMSystem` signaling is to be carried in HLS manifests, both a `<cpix:HLSSignalingData playlist="media">` and a `<cpix:HLSSignalingData playlist="master">` elements must be included in the CPIX request and response.

<cpix:ContentKeyPeriodList> section

This is an optional section, relevant only for Live streaming, defining the crypto periods applied to the content.

The `<cpix:ContentKeyPeriodList>` element can contain one or several `<cpix:ContentKeyPeriod>` child elements, each of them describing a distinct crypto period in the live timeline. Using UUIDs as part of the value of the id attribute is a commonly used approach.

```
<cpix:ContentKeyPeriodList>
 <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
</cpix:ContentKeyPeriodList>
```

Element supported by SPEKE	Mandatory attributes	Optional attributes	Mandatory child elements	Optional child elements
<cpix:ContentKeyPeriodList>	-	id	at least one <cpix:ContentKeyPeriod>	-
<cpix:ContentKeyPeriod>	id, index	-	-	-

If crypto periods are used, the encryption keys also need to be attached to one of the crypto periods in the CPIX document, as shown in the section below.

<cpix:ContentKeyUsageRuleList> section

This is a mandatory section, relevant for both Live and VOD streaming, defining how the different content keys will protect tracks inside the streamset and across the crypto periods.

The <cpix:ContentKeyUsageRuleList> element can contain one or several <cpix:ContentKeyUsageRule> child elements, each of them describing the tracks to which a given content key is applied by the encryptor, potentially during a specific crypto period. At least one <cpix:AudioFilter> or one <cpix:VideoFilter> element is required to be present in a <cpix:ContentKeyUsageRule> element.

The following example shows a simple list with only one rule applying a single content key to all audio and video tracks during a specific crypto period.

```
<cpix:ContentKeyUsageRuleList>
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="ALL">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
  <cpix:VideoFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Element supported by SPEKE	Mandatory attributes	Optional attributes	Mandatory child elements	Optional child elements
<cpix:ContentKeyUsageRuleList>	-	id	at least one <cpix:ContentKeyUsageRule>	-
<cpix:ContentKeyUsageRule>	kid, intendedTrackType	-	at least one <cpix:AudioFilter> or one <cpix:VideoFilter> (*)	<cpix:KeyPeriodFilter>
<cpix:KeyPeriodFilter>	periodId	-	-	-
<cpix:AudioFilter>	-	minChannels, maxChannels	-	-

Element supported by SPEKE	Mandatory attributes	Optional attributes	Mandatory child elements	Optional child elements
<cpix:VideoFilter>	-	minPixels, maxPixels, hdr, minFps, maxFps	-	-

(*) For a detailed explanation on the use of single or of multiple content keys to protect one or several tracks in a streamset, please refer to the Encryption Contract (p. 28) documentation section._

SPEKE API v2 - Encryption contract

The encryption contract defines which content keys are protecting which tracks inside a given streamset, based on the tracks characteristics.

Using multiple content keys for different tracks in a streamset, despite being a recommended industry best practice, is not mandatory, but recommended - at least two different content keys, one for audio tracks and one for video tracks. Using a single content key to encrypt multiple tracks is possible but needs to be explicitly signaled in the CPIX document sent by the encryptor to to key provider. Generally speaking, the encryptor always describes precisely how many content keys are required and how they are leveraged to encrypt the various media tracks.

Principles

The encryption contract is located in the `<cpix:ContentKeyUsageRuleList>` section of the CPIX document. In this section, each content key defined in the `<cpix:ContentKeyList>` section corresponds to a specific `<cpix:ContentKeyUsageRule>` element, which shall include:

- a `ContentKeyUsageRule@intendedTrackType` attribute that can reference one or more sub-components, separated by the '+' sign if multiple sub-components are used. The value of `ContentKeyUsageRule@intendedTrackType` shall be unique in an encryption contract, and can not be used in multiple `ContentKeyUsageRule` elements.
- one or more `<cpix:AudioFilter>` or `<cpix:VideoFilter>` child element, depending on the value of `ContentKeyUsageRule@intendedTrackType` attribute.

The rules governing this relationship are the following:

- When all the audio and video tracks of the streamset need to be protected with a unique content key, the string `'ALL'` must be used as the `ContentKeyUsageRule@intendedTrackType` attribute value. Example 1 shows such a use case. In this situation, both a `<cpix:AudioFilter />` and a `<cpix:VideoFilter />` child elements without any attribute shall be included. Any other combination of `<cpix:AudioFilter>` and/or `<cpix:VideoFilter>` elements is invalid in this particular context.
- For all other use cases, the value of the `ContentKeyUsageRule@intendedTrackType` attribute can be freely defined, and the number of `<cpix:AudioFilter />` and a `<cpix:VideoFilter />` child elements must correspond to the number of sub-components aggregated through the '+' sign. Examples 2/3/4/5/6/7/9/10 illustrate this requirement, when a single sub-component is present in the `ContentKeyUsageRule@intendedTrackType` attribute value. Example 8 illustrate it when multiple sub-components are used: `ContentKeyUsageRule@intendedTrackType="SD+HD"` is described by two distinct `<cpix:VideoFilter>` child elements with different attributes values, and `ContentKeyUsageRule@intendedTrackType="HDR+HFR+UHD"` is described by three distinct `<cpix:VideoFilter>` child elements with different attributes values.

Filters

CPIX defines multiple filtering elements and attributes, but SPEKE supports only a subset of it. The following table summarizes these differences:

CPIX filter type	Overall SPEKE support	Filter attributes supported by SPEKE	Filter attributes not supported by SPEKE
<cpix:VideoFilter>	Yes	minPixels, maxPixels, hdr, minFps, maxFps (optional attributes)	wcg
<cpix:AudioFilter>	Yes	minChannels, maxChannels (optional attributes)	
<cpix:KeyPeriodFilter>	Yes	periodId (mandatory attribute)	
<cpix:BitrateFilter>	No	N/A	N/A
<cpix:LabelFilter>	No	N/A	N/A

As per the CPIX specification for VideoFilter, [minPixels, maxPixels] is an all inclusive range in both dimensions, while (minFps, maxFps] is inclusive only for the maxFps dimension. For AudioFilter, [minChannels, maxChannels] is an inclusive range in both dimensions.

Problematic situations

There are situations where the information provided in the encryption contract might be partial, ambiguous or erroneous. In these cases, it is important that the encryptor and the key provider behave appropriately and guarantee a proper protection of the contents. The following table presents the recommended behavior in these situations:

In this situation	The encryptor should/shall ...	The key provider should/shall ...
No rule applies to one or more tracks in the streamset (see example 3 below)	The encryptor should look at its configuration (external to the CPIX payload) and verify that the concerned tracks don't require encryption. If it's not the expectation, the encryptor should throw an error and stop processing.	*Not relevant: the key provider doesn't have knowledge of the streamset structure.*
Multiple rules overlap and suggest multiple content keys to encrypt a specific track	The encryptor should apply the last ContentKeyUsageRule successfully evaluated in the order of the document.	*Not relevant: the key provider doesn't have knowledge of the streamset structure.*
The encryption contract changes in a single SPEKE request/response cycle	The encryptor shall raise an exception and stop processing, as the key provider is not responsible for defining the encryption contract.	To prevent this situation to happen in the first place, the key provider must not modify an encryption contract received in the CPIX payload of the SPEKE request.
Malformed encryption contract: intendedTrackType/Filters	The encryptor shall raise an exception, stop processing and	The key provider shall raise an exception and return a

In this situation	The encryptor should/shall ...	The key provider should/shall ...
cardinality constraint exception, unsupported filters or attributes	not send the SPEKE request to the key provider, as it would most likely result in erroneous content protection or leave some tracks unprotected.	'Malformed encryption contract' error.
Well formed encryption contract, but in breach of the DRM security levels constraints: as an example, a single content key being requested to protect both audio tracks and UHD video tracks	If the encryptor has got knowledge of the DRM security levels constraints, it should raise an exception, stop processing and not send the SPEKE request to the key provider, as it would most likely result in erroneous content protection.	The key provider shall raise an exception and return a 'Requested CPIX encryption contract not supported' error.
Missing encryption contract	The encryptor shall not send CPIX documents which don't contain any VideoFilter or AudioFilter element.	The key provider shall raise an exception and return a 'Missing CPIX encryption contract' error.

Examples of Encryption Contracts

Example 1: one content key for all audio and video tracks

```
<cpix:ContentKeyUsageRuleList>
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="ALL">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
  <cpix:VideoFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 2: one content key for all video tracks, one content key for all audio tracks

```
<cpix:ContentKeyUsageRuleList>
    <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
        <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
        <cpix:VideoFilter />
    </cpix:ContentKeyUsageRule>
    <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
        <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
        <cpix:AudioFilter />
    </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 3: one content key for all video tracks, unencrypted audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter />
 </cpix:ContentKeyUsageRule>
```

```
</cpix:ContentKeyUsageRuleList>
```

Example 4: multiple content keys for different video tracks (SD/HD), one content key for all audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for SD video tracks (up to 1024x576) -->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="SD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter maxPixels="589824" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for HD video tracks (more than 1024x576) -->
 <cpix:ContentKeyUsageRule kid="37e3de05-9a3b-4c69-8970-63c17a95e0b7"
 intendedTrackType="HD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="589825" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for all audio tracks -->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 5: multiple content keys for different video tracks (SD/HD/UHD), one content key for all audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for SD video tracks (up to 1024x576) -->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="SD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter maxPixels="589824" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for HD video tracks (more than 1024x576, up to 1920x1080) -->
 <cpix:ContentKeyUsageRule kid="37e3de05-9a3b-4c69-8970-63c17a95e0b7"
 intendedTrackType="HD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="589825" maxPixels="2073600" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for UHD video tracks (more than 1920x1080) -->
 <cpix:ContentKeyUsageRule kid="75c6fa78-8b5d-6d75-9653-26f41b78d1a3"
 intendedTrackType="UHD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="2073601" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for all audio tracks -->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 6: multiple content keys for different video tracks (SD/HD/UHD1/UHD2), one content key for all audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for SD video tracks (up to 1024x576) -->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="SD">
```

```
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter maxPixels="589824" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for HD video tracks (more than 1024x576, up to 1920x1080) -->
 <cpix:ContentKeyUsageRule kid="37e3de05-9a3b-4c69-8970-63c17a95e0b7"
 intendedTrackType="HD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="589825" maxPixels="2073600" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for UHD1 video tracks (more than 1920x1080, up to 4096x2160) -->
 <cpix:ContentKeyUsageRule kid="75c6fa78-8b5d-6d75-9653-26f41b78d1a3"
 intendedTrackType="UHD1">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="2073601" maxPixels="8847360" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for UHD2 video tracks (more than 4096x2160) -->
 <cpix:ContentKeyUsageRule kid="63d2ec36-6b7c-9f34-4546-97d01f36f7c5"
 intendedTrackType="UHD2">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="8847361" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for all audio tracks -->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 7: multiple content keys for different video tracks (SD/HD1/HD2/UHD1/UHD2), one content key for all audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for SD video tracks (up to 1024x576) -->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="SD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter maxPixels="589824" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for HD1 video tracks (more than 1024x576, up to 1280x720) -->
 <cpix:ContentKeyUsageRule kid="37e3de05-9a3b-4c69-8970-63c17a95e0b7"
 intendedTrackType="HD1">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="589825" maxPixels="921600" />
 </cpix:ContentKeyUsageRule>
        <!-- Rule for HD2 video tracks (more than 1280x720, up to 1920x1080) -->
          <cpix:ContentKeyUsageRule kid="cda406d8-9d87-4f76-92da-31110e756176"
 intendedTrackType="HD2">
            <cpix:KeyPeriodFilter
 periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
            <cpix:VideoFilter minPixels="921601" maxPixels="2073600" />
          </cpix:ContentKeyUsageRule>
 <!-- Rule for UHD1 video tracks (more than 1920x1080, up to 4096x2160) -->
 <cpix:ContentKeyUsageRule kid="75c6fa78-8b5d-6d75-9653-26f41b78d1a3"
 intendedTrackType="UHD1">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="2073601" maxPixels="8847360" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for UHD2 video tracks (more than 4096x2160) -->
 <cpix:ContentKeyUsageRule kid="63d2ec36-6b7c-9f34-4546-97d01f36f7c5"
 intendedTrackType="UHD2">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter minPixels="8847361" />
 </cpix:ContentKeyUsageRule>
```

```
 <!-- Rule for all audio tracks -->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 8: multiple content keys for different video tracks (based on multiple attributes types), one content key for all audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for SD and HD video tracks-->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff" intendedTrackType="SD
+HD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter maxPixels="442368" maxFps="30" hdr="false"/>
  <cpix:VideoFilter minPixels="442369" maxPixels="2073600" maxFps="30" hdr="false"/>
 </cpix:ContentKeyUsageRule>
 <!-- Rule for HDR, HFR and UHD video tracks-->
 <cpix:ContentKeyUsageRule kid="37e3de05-9a3b-4c69-8970-63c17a95e0b7"
 intendedTrackType="HDR+HFR+UHD">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter hdr="true" />
  <cpix:VideoFilter minFps="30" />
  <cpix:VideoFilter minPixels="20736001" />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for all audio tracks-->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter />
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 9: one content keys for all video tracks, multiple content keys for stereo and multichannel audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for video tracks-->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for stereo audio tracks-->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="STEREO_AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter maxChannels="2"/>
 </cpix:ContentKeyUsageRule>
 <!-- Rule for multichannel audio tracks-->
 <cpix:ContentKeyUsageRule kid="7ae8e96f-309e-42c3-a510-24023d923373"
 intendedTrackType="MULTICHANNEL_AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <AudioFilter minChannels="3"/>
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

Example 10: one content keys for all video tracks, multiple content keys for stereo and two types of multichannel audio tracks

```
<cpix:ContentKeyUsageRuleList>
 <!-- Rule for video tracks-->
 <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:VideoFilter />
 </cpix:ContentKeyUsageRule>
 <!-- Rule for stereo audio tracks-->
 <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="STEREO_AUDIO">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter maxChannels="2"/>
 </cpix:ContentKeyUsageRule>
 <!-- Rule for multichannel audio tracks (3 to 6 channels)-->
 <cpix:ContentKeyUsageRule kid="7ae8e96f-309e-42c3-a510-24023d923373"
 intendedTrackType="MULTICHANNEL_AUDIO_3_6">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter minChannels="3" maxChannels="6"/>
 </cpix:ContentKeyUsageRule>
  <!-- Rule for multichannel audio tracks (7 channels and more)-->
 <cpix:ContentKeyUsageRule kid="81eb3761-55ff-4d22-a31d-94f01bbfd8ba"
 intendedTrackType="MULTICHANNEL_AUDIO_7">
  <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
  <cpix:AudioFilter minChannels="7"/>
 </cpix:ContentKeyUsageRule>
</cpix:ContentKeyUsageRuleList>
```

SPEKE API v2 - Live workflow method call examples

Request Syntax Example

The following URL is an example and does not indicate a fixed format:

```
POST https://speke-compatible-server/speke/v2.0/copyProtection
```

Request Body

A CPIX document.

Request Headers

Name	Type	Occurs	Description
`AWS Authorization`	String	1..1	See AWS Sigv4
`X-Amz-Security-Token`	String	1..1	See AWS Sigv4
`X-Amz-Date`	String	1..1	See AWS Sigv4
`Content-Type`	String	1..1	application/xml
`X-Speke-Version`	String	1..1	SPEKE API version used with the request, formulated as MajorVersion.MinorVersion, like '2.0' for SPEKE v2.0

Response Headers

Name	Type	Occurs	Description
`X-Speke-User-Agent`	String	1..1	String that identifies the key provider
`Content-Type`	String	1..1	application/xml

Request Response

HTTP CODE	Payload Name	Occurs	Description
`200 (Success)`	CPIX	1..1	DASH-CPIX payload response
`4XX (Client error)`	Client error message	1..1	Description of the client error
`5XX (Server error)`	Server error message	1..1	Description of the server error

Note
The examples in this section do not include content key encryption. For information on how to add content key encryption, see Content key encryption (p. 41).

Live Example Request Payload with Keys in the Clear

The following example shows a typical live request payload from the encryptor to the DRM key provider, with one content key for all video tracks and one content key for all audio tracks:

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs"></cpix:ContentKey>
  <cpix:ContentKey explicitIV="L6jzdXrXAFbCJGBuMrrKrG==" kid="53abdba2-f210-43cb-bc90-
f18f9a890a02" commonEncryptionScheme="cbcs"></cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- FairPlay -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <!-- Widevine -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
```

```
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
  <!-- Playready -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData></cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData></cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyPeriodList>
  <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
 </cpix:ContentKeyPeriodList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
   <cpix:VideoFilter />
  </cpix:ContentKeyUsageRule>
  <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
   <cpix:AudioFilter />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

Live Example Response Payload with Keys in the Clear

The following example shows a typical response payload from the DRM key provider (returned values have been shortened with [...] for readability):

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
  <cpix:ContentKey explicitIV="L6jzdXrXAFbCJGBuMrrKrG==" kid="53abdba2-f210-43cb-bc90-
f18f9a890a02" commonEncryptionScheme="cbcs">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>h3toSFIlyAYpfXVQ795m6x==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
```

```
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- FairPlay -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media">aHR0cHM6L[...]WZm</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">Y29tLmFwc[...]XJ5</cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media">trBAnbMcj[...]u44</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">mn626PjyR[...]2fi</cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <!-- Widevine -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media">Ifa2V5LWl[...]nNB</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">oIARIQeSI[...]Nd2l</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>RoNd2lkZXZ[...]Nib</cpix:ContentProtectionData>
   <cpix:PSSH>AAAAanBzc[...]A==</cpix:PSSH>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media">lTznjvtzL[...]GfJ</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">XgzdzQH7p[...]zeX</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>TdgRnuJsZ[...]wDw</cpix:ContentProtectionData>
   <cpix:PSSH>mYZbjpWdS[...]D==</cpix:PSSH>
  </cpix:DRMSystem>
  <!-- Playready -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media">HicXmbZ2m[...]4==</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">GVzdCIfa2[...]Eta</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>t7WwH24FI[...]YCC</cpix:ContentProtectionData>
   <cpix:PSSH>FFFFanBzc[...]A==</cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData>s5RrJ12HL[...]UBB</
cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media">BptGzwis2[...]Iej</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">3c9SXdVa0[...]MBH</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>HotJCMQyc[...]GpU</cpix:ContentProtectionData>
   <cpix:PSSH>S6UD43ybN[...]f==</cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData>VBFUv2or0[...]JeP</
cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyPeriodList>
  <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
 </cpix:ContentKeyPeriodList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
   <cpix:VideoFilter />
  </cpix:ContentKeyUsageRule>
  <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
   <cpix:AudioFilter />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

SPEKE API v2 - VOD workflow method call examples

Request Syntax Example

The following URL is an example and does not indicate a fixed format.

```
POST https://speke-compatible-server/speke/v2.0/copyProtection
```

Request Body

A CPIX document.

Request Headers

Name	Type	Occurs	Description
`AWS Authorization`	String	1..1	See AWS Sigv4
`X-Amz-Security-Token`	String	1..1	See AWS Sigv4
`X-Amz-Date`	String	1..1	See AWS Sigv4
`Content-Type`	String	1..1	application/xml
`X-Speke-Version`	String	1..1	SPEKE API version used with the request, formulated as MajorVersion.MinorVersion, like '2.0' for SPEKE v2.0

Response Headers

Name	Type	Occurs	Description
`X-Speke-User-Agent`	String	1..1	String that identifies the key provider
`Content-Type`	String	1..1	application/xml

Request Response

HTTP CODE	Payload Name	Occurs	Description
`200 (Success)`	CPIX	1..1	DASH-CPIX payload response
`4XX (Client error)`	Client error message	1..1	Description of the client error
`5XX (Server error)`	Server error message	1..1	Description of the server error

Note
The examples in this section do not include content key encryption. For information on how to add content key encryption, see Content key encryption (p. 41).

VOD Example Request Payload with Keys in the Clear

The following example shows a typical VOD request payload from the encryptor to the DRM key provider, with one content key for all video tracks and one content key for all audio tracks:

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs"></cpix:ContentKey>
  <cpix:ContentKey explicitIV="L6jzdXrXAFbCJGBuMrrKrG==" kid="53abdba2-f210-43cb-bc90-
f18f9a890a02" commonEncryptionScheme="cbcs"></cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- FairPlay -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <!-- Widevine -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
  <!-- Playready -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData></cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData></cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
```

```
   <cpix:VideoFilter />
  </cpix:ContentKeyUsageRule>
  <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
   <cpix:AudioFilter />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

VOD Example Response Payload with Keys in the Clear

The following example shows a typical response payload from the DRM key provider (returned values have been shortened with [...] for readability):

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
  <cpix:ContentKey explicitIV="L6jzdXrXAFbCJGBuMrrKrG==" kid="53abdba2-f210-43cb-bc90-
f18f9a890a02" commonEncryptionScheme="cbcs">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>h3toSFIlyAYpfXVQ795m6x==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- FairPlay -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media">aHR0cHM6L[...]WZm</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">Y29tLmFwc[...]XJ5</cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="94ce86fb-07ff-4f43-
adb8-93d2fa968ca2">
   <cpix:HLSSignalingData playlist="media">trBAnbMcj[...]u44</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">mn626PjyR[...]2fi</cpix:HLSSignalingData>
  </cpix:DRMSystem>
  <!-- Widevine -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media">Ifa2V5LWl[...]nNB</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">oIARIQeSI[...]Nd2l</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>RoNd2lkZXZ[...]Nib</cpix:ContentProtectionData>
   <cpix:PSSH>AAAAanBzc[...]A==</cpix:PSSH>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media">lTznjvtzL[...]GfJ</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">XgzdzQH7p[...]zeX</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>TdgRnuJsZ[...]wDw</cpix:ContentProtectionData>
   <cpix:PSSH>mYZbjpWdS[...]D==</cpix:PSSH>
  </cpix:DRMSystem>
  <!-- Playready -->
  <cpix:DRMSystem kid="98ee5596-cd3e-a20d-163a-e382420c6eff" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
```

```
   <cpix:HLSSignalingData playlist="media">HicXmbZ2m[...]4==</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">GVzdCIfa2[...]Eta</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>t7WwH24FI[...]YCC</cpix:ContentProtectionData>
   <cpix:PSSH>FFFFanBzc[...]A==</cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData>s5RrJ12HL[...]UBB</
cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
  <cpix:DRMSystem kid="53abdba2-f210-43cb-bc90-f18f9a890a02" systemId="9a04f079-9840-4286-
ab92-e65be0885f95">
   <cpix:HLSSignalingData playlist="media">BptGzwis2[...]Iej</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">3c9SXdVa0[...]MBH</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>HotJCMQyc[...]GpU</cpix:ContentProtectionData>
   <cpix:PSSH>S6UD43ybN[...]f==</cpix:PSSH>
   <cpix:SmoothStreamingProtectionHeaderData>VBFUv2or0[...]JeP</
cpix:SmoothStreamingProtectionHeaderData>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="98ee5596-cd3e-a20d-163a-e382420c6eff"
 intendedTrackType="VIDEO">
   <cpix:VideoFilter />
  </cpix:ContentKeyUsageRule>
  <cpix:ContentKeyUsageRule kid="53abdba2-f210-43cb-bc90-f18f9a890a02"
 intendedTrackType="AUDIO">
   <cpix:AudioFilter />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

SPEKE API v2 - Content key encryption

You can optionally add content key encryption to your SPEKE implementation. Content key encryption guarantees full end-to-end protection by encrypting the content keys for transit, in addition to encrypting the content itself. If you don't implement this for your key provider, you rely on the transport layer encryption plus strong authentication for security.

To use content key encryption for encryptors running in AWS Cloud, customers import certificates into the AWS Certificate Manager and then use the resulting certificate ARNs for their encryption activities. The encryptor uses the certificate ARNs and the ACM service to provide encrypted content keys to the DRM key provider.

Restrictions

SPEKE supports content key encryption as specified in the DASH-IF CPIX specification with the following restrictions:

- SPEKE doesn't support digital signature verification (XMLDSIG) for request or response payloads.
- SPEKE requires 2048 RSA-based certificates.

These restrictions are also listed in Customizations and constraints to the DASH-IF specification (p. 22).

Implement content key encryption

To provide content key encryption, include the following in your DRM key provider implementations:

- Handle the element `<cpix:DeliveryDataList>` in the request and response payloads.
- Provide encrypted values in the `<cpix:ContentKeyList>` of the response payloads.

For more information about these elements, see the DASH-IF CPIX 2.3 specification.

Example Content Key Encryption Element `<cpix:DeliveryDataList>` *in the Request Payload*

```
<cpix:CPIX contentId="abc123"
    version="2.3"
    xmlns:cpix="urn:dashif:org:cpix"
    xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
    <cpix:DeliveryDataList>
        <cpix:DeliveryData id="<ORIGIN SERVER ID>">
            <cpix:DeliveryKey>
                <ds:X509Data>
                    <ds:X509Certificate><X.509 CERTIFICATE, BASE-64 ENCODED></
ds:X509Certificate>
                </ds:X509Data>
            </cpix:DeliveryKey>
        </cpix:DeliveryData>
    </cpix:DeliveryDataList>
    <cpix:ContentKeyList>
     ...
    </cpix:ContentKeyList>
</cpix:CPIX>
```

Example Content Key Encryption Element `<cpix:DeliveryDataList>` *in the Response Payload*

```
<cpix:CPIX contentId="abc123"
    version="2.3"
    xmlns:cpix="urn:dashif:org:cpix"
    xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
    <cpix:DeliveryDataList>
        <cpix:DeliveryData id="<ORIGIN SERVER ID>">
            <cpix:DeliveryKey>
                <ds:X509Data>
                    <ds:X509Certificate><X.509 CERTIFICATE, BASE-64 ENCODED></
ds:X509Certificate>
                </ds:X509Data>
            </cpix:DeliveryKey>
            <cpix:DocumentKey Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc">
                <cpix:Data>
                    <pskc:Secret>
                        <pskc:EncryptedValue>
                            <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#rsa-oaep-mgf1p" />
                            <enc:CipherData>
                                <enc:CipherValue><RSA CIPHER VALUE></enc:CipherValue>
                            </enc:CipherData>
                        </pskc:EncryptedValue>
                        <pskc:ValueMAC>qnei/5TsfUwDu+8bhsZrLjDRDngvmnUZD2eva7SfXWw=</
pskc:ValueMAC>
                    </pskc:Secret>
                </cpix:Data>
            </cpix:DocumentKey>
            <cpix:MACMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha512">
                <cpix:Key>
                    <pskc:EncryptedValue>
                        <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#rsa-oaep-mgf1p" />
                        <enc:CipherData>
                            <enc:CipherValue><RSA CIPHER VALUE></enc:CipherValue>
                        </enc:CipherData>
                    </pskc:EncryptedValue>
                    <pskc:ValueMAC>DGqdpHUfFKxdsO9+EWrPjtdTCVfjPLwwtzEcFC/j0xY=</
pskc:ValueMAC>
                </cpix:Key>
            </cpix:MACMethod>
        </cpix:DeliveryData>
```

```
    </cpix:DeliveryDataList>
    <cpix:ContentKeyList>
     ...
    </cpix:ContentKeyList>
</cpix:CPIX>
```

Example Content Key Encryption Element `<cpix:ContentKeyList>` *in the Response Payload*

The following example shows encrypted content key handling in the `<cpix:ContentKeyList>` element of the response payload. This uses the `<pskc:EncryptedValue>` element:

```
<cpix:ContentKeyList>
     <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs">
         <cpix:Data>
             <pskc:Secret>
                 <pskc:EncryptedValue>
                     <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#aes256-cbc" />
                     <enc:CipherData>
                         <enc:CipherValue>NJYebfvJ2TdMm3k6v
+rLNVYb0NoTJoTLBBdbpe8nmilEfp82SKa7MkqTn2lmQBPB</enc:CipherValue>
                     </enc:CipherData>
                 </pskc:EncryptedValue>
                 <pskc:ValueMAC>t9lW4WCebfS1GP+dh0IicMs+2+jnrAmfDa4WU6VGHc4=</
pskc:ValueMAC>
             </pskc:Secret>
         </cpix:Data>
     </cpix:ContentKey>
 </cpix:ContentKeyList>
```

By comparison, the following example shows a similar response payload with the content key delivered unencrypted, as a clear key. This uses the `<pskc:PlainValue>` element:

```
<cpix:ContentKeyList>
    <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw==" kid="98ee5596-cd3e-a20d-163a-
e382420c6eff" commonEncryptionScheme="cbcs">
        <cpix:Data>
            <pskc:Secret>
                <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
            </pskc:Secret>
        </cpix:Data>
    </cpix:ContentKey>
</cpix:ContentKeyList>
```

SPEKE API v2 - Overriding the key identifier

The encryptor creates a new key identifier (KID) each time that it rotates keys. It passes the KID to the DRM key provider in its requests. Almost always, the key provider responds using the same KID, but it can provide a different value for the KID in the response.

The following is an example request with the KID `11111111-1111-1111-1111-111111111111`:

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw=="
 kid="11111111-1111-1111-1111-111111111111" commonEncryptionScheme="cbcs"></
cpix:ContentKey>
```

```
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- Widevine -->
  <cpix:DRMSystem kid="11111111-1111-1111-1111-111111111111" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media"></cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master"></cpix:HLSSignalingData>
   <cpix:ContentProtectionData></cpix:ContentProtectionData>
   <cpix:PSSH></cpix:PSSH>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyPeriodList>
  <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
 </cpix:ContentKeyPeriodList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="11111111-1111-1111-1111-111111111111"
 intendedTrackType="VIDEO">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
   <cpix:VideoFilter />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

The following response overrides the KID to `22222222-2222-2222-2222-222222222222`:

```
<cpix:CPIX contentId="abc123" version="2.3" xmlns:cpix="urn:dashif:org:cpix"
 xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
 <cpix:ContentKeyList>
  <cpix:ContentKey explicitIV="OFj2IjCsPJFfMAxmQxLGPw=="
 kid="22222222-2222-2222-2222-222222222222" commonEncryptionScheme="cbcs">
   <cpix:Data>
    <pskc:Secret>
     <pskc:PlainValue>5dGAgwGuUYu4dHeHtNlxJw==</pskc:PlainValue>
    </pskc:Secret>
   </cpix:Data>
  </cpix:ContentKey>
 </cpix:ContentKeyList>
 <cpix:DRMSystemList>
  <!-- Widevine -->
  <cpix:DRMSystem kid="22222222-2222-2222-2222-222222222222" systemId="edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed">
   <cpix:HLSSignalingData playlist="media">Ifa2V5LWl[...]nNB</cpix:HLSSignalingData>
   <cpix:HLSSignalingData playlist="master">oIARIQeSI[...]Nd2l</cpix:HLSSignalingData>
   <cpix:ContentProtectionData>RoNd2lkZXZ[...]Nib</cpix:ContentProtectionData>
   <cpix:PSSH>AAAAanBzc[...]A==</cpix:PSSH>
  </cpix:DRMSystem>
 </cpix:DRMSystemList>
 <cpix:ContentKeyPeriodList>
  <cpix:ContentKeyPeriod id="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f" index="1" />
 </cpix:ContentKeyPeriodList>
 <cpix:ContentKeyUsageRuleList>
  <cpix:ContentKeyUsageRule kid="22222222-2222-2222-2222-222222222222"
 intendedTrackType="VIDEO">
   <cpix:KeyPeriodFilter periodId="keyPeriod_0909829f-40ff-4625-90fa-75da3e53278f"/>
   <cpix:VideoFilter />
  </cpix:ContentKeyUsageRule>
 </cpix:ContentKeyUsageRuleList>
</cpix:CPIX>
```

License

Creative Commons Attribution-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. BY-SA Compatible License means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

d. Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

e. Effective Technological Measures means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

f. Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

g. License Elements means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution and ShareAlike.

h. Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

i. Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

j. Licensor means the individual(s) or entity(ies) granting rights under this Public License.

k. Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

l. Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

m. You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

   A. reproduce and Share the Licensed Material, in whole or in part; and

   B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

   A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

   B. Additional offer from the Licensor – Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

   C. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

   A. retain the following if it is supplied by the Licensor with the Licensed Material:

```
i . identification of the creator(s) of the Licensed Material and any others
 designated to receive attribution, in any reasonable manner requested by the
 Licensor (including by pseudonym if designated);
```

```
ii . a copyright notice;
```

```
iii . a notice that refers to this Public License;
```

```
iv . a notice that refers to the disclaimer of warranties;
```

```
v . a URI or hyperlink to the Licensed Material to the extent reasonably
 practicable;
```

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

b. ShareAlike. In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

1. The Adapter's License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-SA Compatible License.

2. You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.

3. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database. For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

   1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

   2. upon express reinstatement by the Licensor.

c. For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

d. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

e. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Document history

The following table describes the changes to the SPEKE documentation.

SPEKE v1

Change	Description	Date
Support matrix - AWS Elemental Delta	Added an AWS Elemental Delta support matrix.	February 7, 2019
Updates to DRM platform providers	Added links and new partner information to the DRM platform provider list.	January 24, 2019
Include third-party encryptors	Updated the architecture and descriptions to account for third-party encryptors.	November 20, 2018
Content Key Encryption	Added the option to encrypt content keys. Prior to this, Secure Packager and Encoder Key Exchange supported clear key delivery only.	October 30, 2018
Support matrix - AWS Elemental Live	Added an AWS Elemental Live support matrix.	September 27, 2018
Standard payload components	Added a section that defines the main elements in the JSON payload.	September 27, 2018
KID override	Added a section about KID overrides by a key provider.	September 27, 2018
Corrected links to DASH-IF site	Corrected links to the DASH IF site for the CPIX specification and for the system IDs page.	September 27, 2018
Release copy for AWS Elemental Live	Updated the SPEKE documentation to include AWS Elemental products.	July 20, 2018
CMAF	Updated the support matrix tables for services to include the Common Media Application Format (CMAF).	June 27, 2018
Initial release	Initial release of Secure Packager and Encoder Key Exchange (SPEKE) version 1, a specification for communication between a content encryptor and a DRM key provider. The DRM key provider exposes a Secure Packager and Encoder Key Exchange API to handle incoming key requests.	November 27, 2017

SPEKE v2

Change	Description	Date
Updates to DRM platform providers and Encryption contract section	Added new qualified partners to the SPEKE v2 column of the DRM platform provider list. Added two new examples of Encryption contracts, and changed SD max resolution to 1024x576 in all concerned examples.	January 27, 2022
Initial release	Initial release of Secure Packager and Encoder Key Exchange (SPEKE) version 2.0, a specification for communication between a content encryptor and a DRM key provider. The DRM key provider exposes a Secure Packager and Encoder Key Exchange API to handle incoming key requests.	September 7, 2021

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.